Estimation of Dynamic Discrete Demand Model with Bayesian Markov Chain Monte Carlo

Bayesian Estimation of Dynamic Discrete Choice Models
Author(s): Susumu Imai, Neelam Jain and Andrew Ching
Source: *Econometrica*, Vol. 77, No. 6 (Nov., 2009), pp. 1865-1899
Published by: The Econometric Society

Raymond Chiasson
RSM3055
ICJ Method Assignment
8-Nov-2022

Summary

I solve the Loyalty Program model using the IJC method in Julia. These slides briefly document the model, method, code, and estimation results.

This project features

- Bayesian Markov Chain Monte Carlo (MCMC), which is an efficient method to estimate structural models with many parameters such as those with heterogeneous individuals.
- The "IJC method" described in Imai, Jain and Ching 2009 which is a computationally efficient way to approximate the value function in dynamic models which are being estimated with Bayesian MCMC.
- The "Loyalty Program" which is a basic discrete dynamic choice model where consumers decide between products in the current period, and can earn "Rewards" after having chose a product a certain number of times (for example, a "buy 5 get one free").
- Clean and optimized Julia code using best practices of the language, to the best of my knowledge.

Model Parameters and Environmental Variables

```
# ------------------------------------------------------------------------------
# 0. Parameters and setup.
# ------------------------------------------------------------------------------
# Model parameters (to be esimated).
α   = [-0.0, -0.0, -0.0]          # store brand intercepts
γ   = -1.0                        # price coefficient
G   = [1.0, 3.0, 8.0]             # value of gifts
σg1 = [0.0, 0.0, 0.0]             # homogeneous consumers
β   = 0.650                       # discount rate

# Environment variables.
n_stores = 3                      # number of stores
n_choices = n_stores + 1          # number of choices for consumer (stores + outside option)
s̄ = [4, 4, 6]                     # gift threshold
price_mean = [1.0, 0.75, 1.5]     # mean of observed prices
price_stdev = [0.25 0.00 0.00;
               0.00 0.25 0.00;
               0.00 0.00 0.25]    # standard deviation (covariance matrix) of observed prices
n_price_draws = 100               # number of draws for price integration

# Simulated data parameters.
n_individuals = 1000
n_periods = 100
```

Parameter Estimates – Bayesian MCMC with IJC

Bayesian MCMC with IJC Results

8x5 DataFrame

Row	variable Any	true value Any	estimated mean Any	95% Credible Interval Low Any	95% Credible Interval High Any
1	α_1	-0.0	-0.0106112	-0.0542716	0.0367328
2	α_2	-0.0	-0.0207863	-0.0559575	0.0169452
3	α_3	-0.0	0.00135663	-0.0615654	0.0694388
4	γ	-1.0	-0.997628	-1.04115	-0.957486
5	G_1	0.65	0.654841	0.641717	0.667859
6	G_2	1.0	1.01293	0.971786	1.05542
7	G_3	3.0	3.1305	3.07501	3.18438
8	β	8.0	7.89058	7.70638	8.04987

```
completed 30000 draws, acceptance rate 0.24567408875385913
```

Parameter Estimates – Maximum Likelihood

Numerical Maximum Likelihood Results
8x5 DataFrame

Row	values Any	true value Any	estimated mean Any	95% Confidence Interval Low Any	95% Condifence Interval High Any
1	α_1	-0.0	-0.0061304	-0.0491608	0.0369
2	α_2	-0.0	0.000857478	-0.0365915	0.0383065
3	α_3	-0.0	-0.000520678	-0.0618666	0.0608253
4	γ	-1.0	-0.998529	-1.03678	-0.960279
5	G_1	1.0	0.995541	0.930543	1.06054
6	G_2	3.0	3.02741	2.93073	3.12408
7	G_3	8.0	7.9929	7.83024	8.15557
8	β	0.65	0.650514	0.639755	0.661272

Initial Parameter Guesses

Variable	Parameter Guess			
	True Value	Informed	Directional	Zeros
α_1	0	0	1	0
α_2	0	0	1	0
α_3	0	0	1	0
γ	-1	-0.5	-1	0
β	0.65	0.8	0.5	0
G_1	1	2	1	0
G_2	3	5	1	0
G_3	8	5	1	0

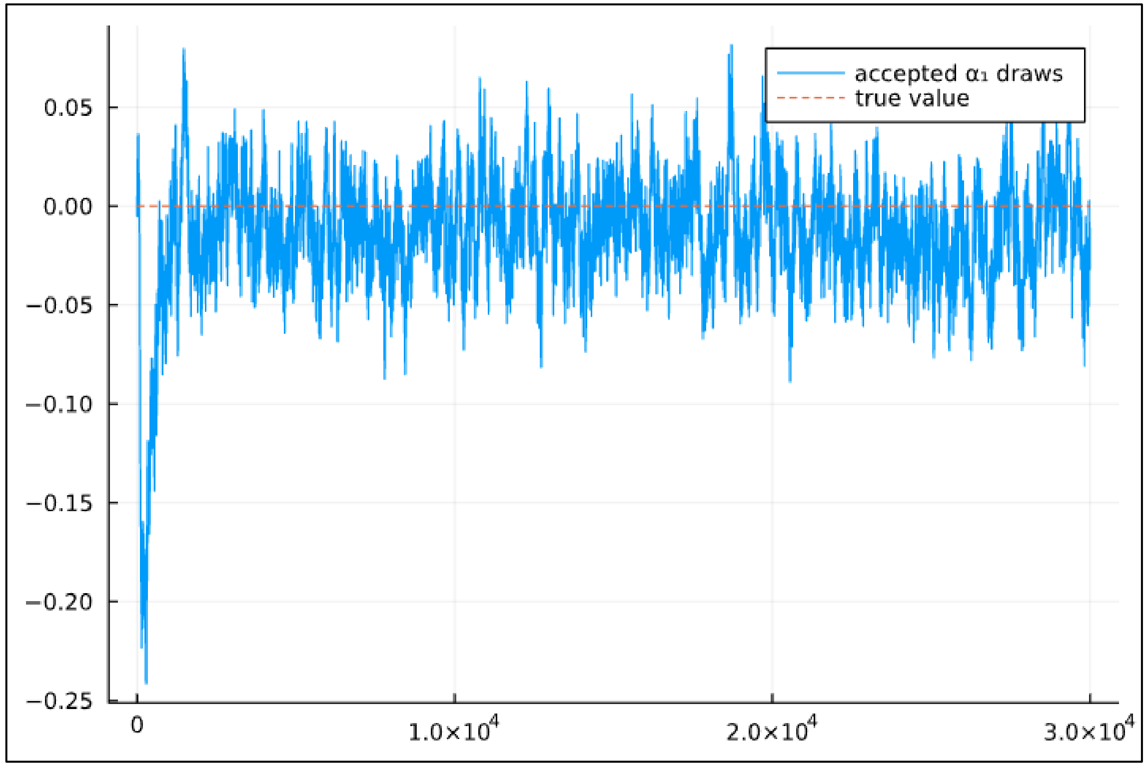
I experimented with different starting parameter values.

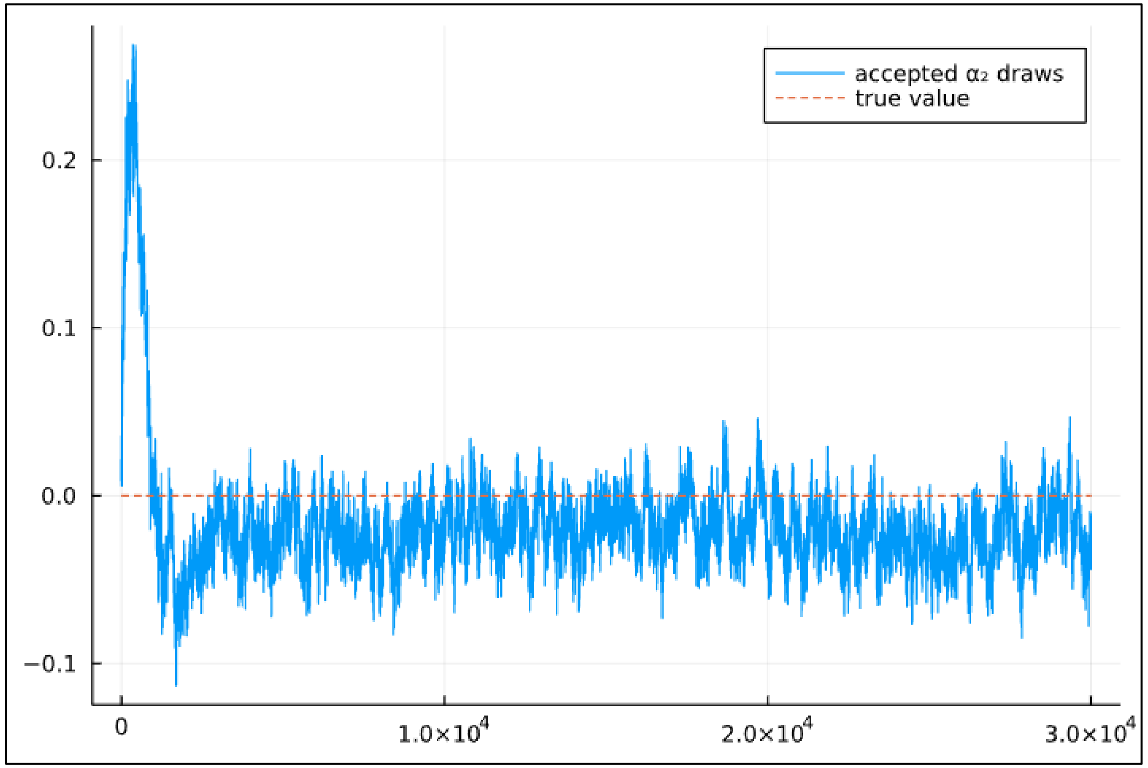
- Starting from the true value, to confirm that the distribution converged around these values.
- Informed and directional guesses which were logical choices based on economic theory and preliminary runs.
- A naïve guess of 0 for each parameter.

The distribution converged well before the burn-in period for all parameter guesses.

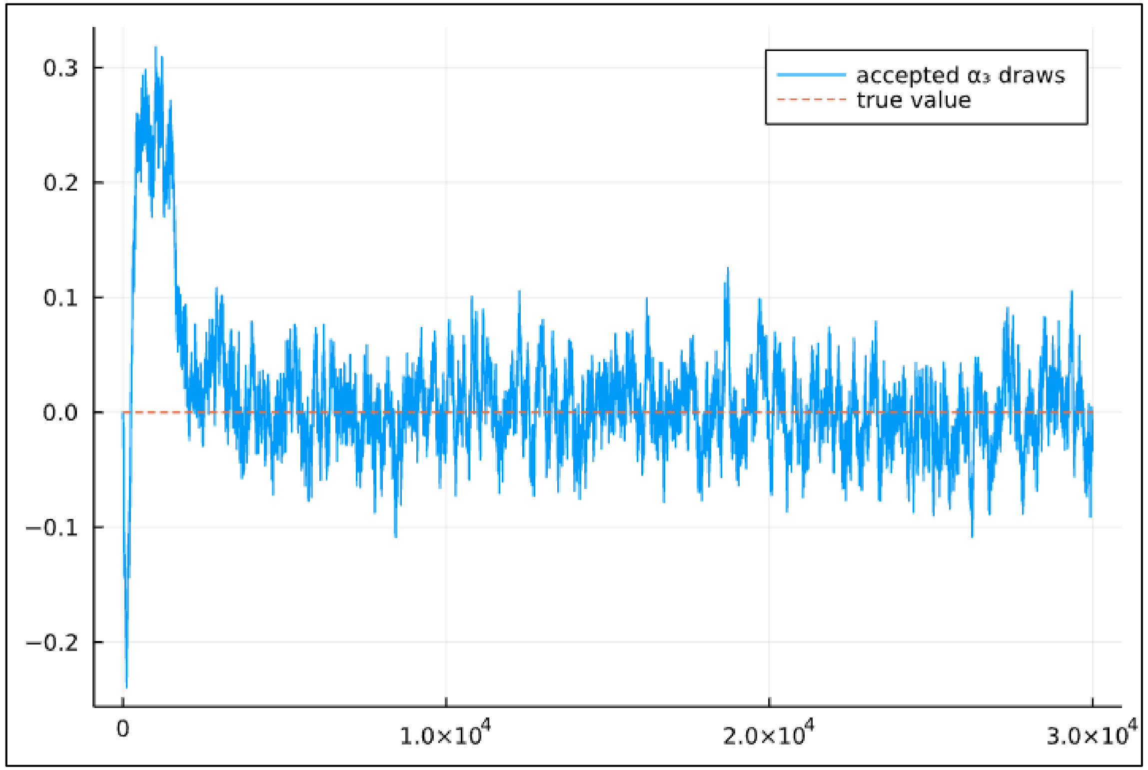
The following graphs and results all use the Zeros parameter guess.

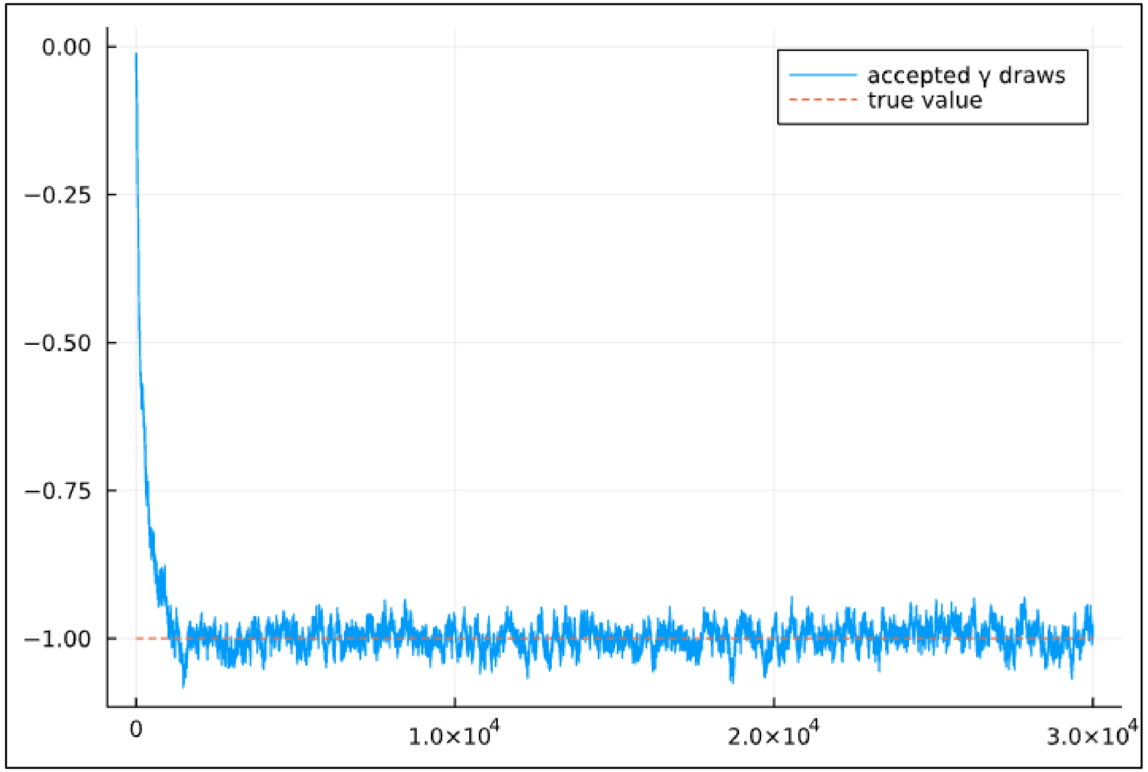
Accepted Parameter Draws from Bayesian Posterior



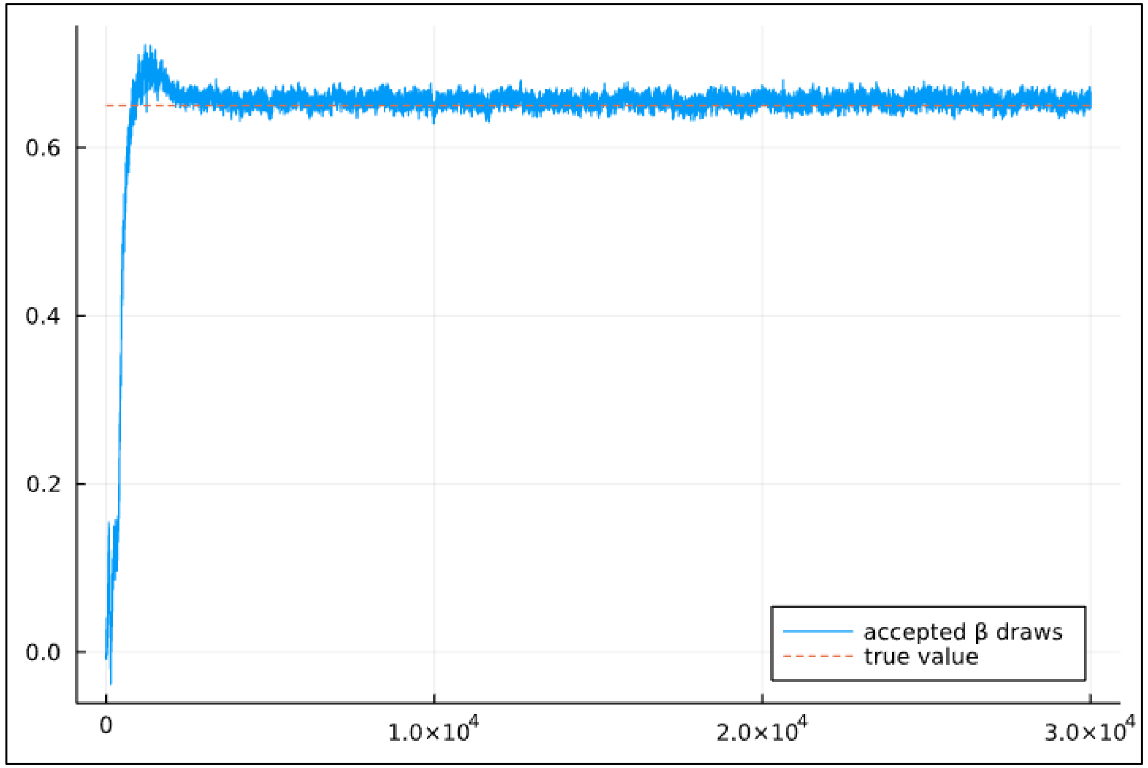


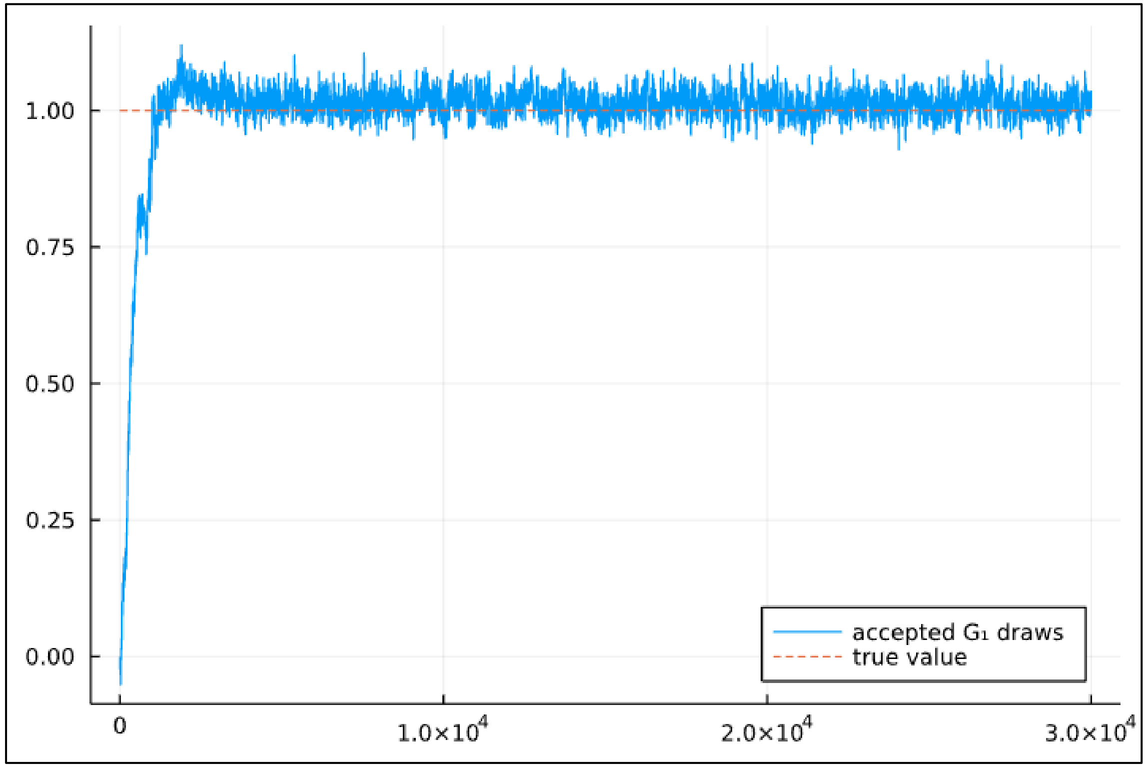
Accepted Parameter Draws from Bayesian Posterior



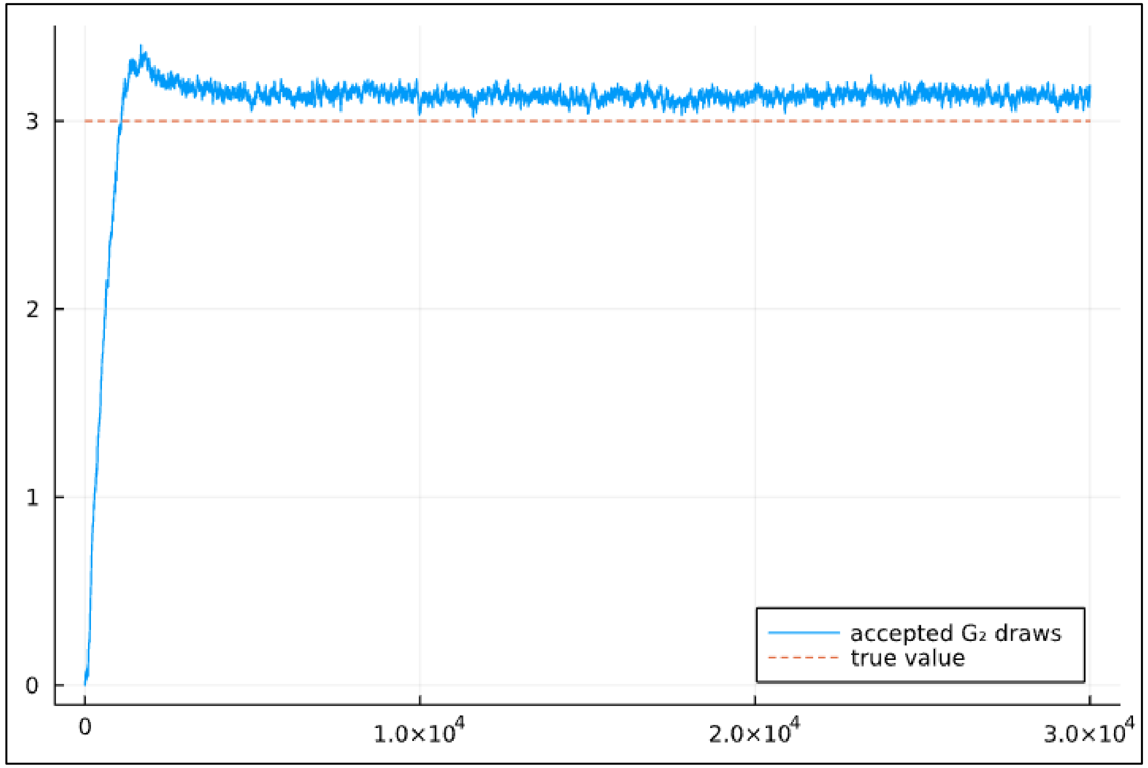


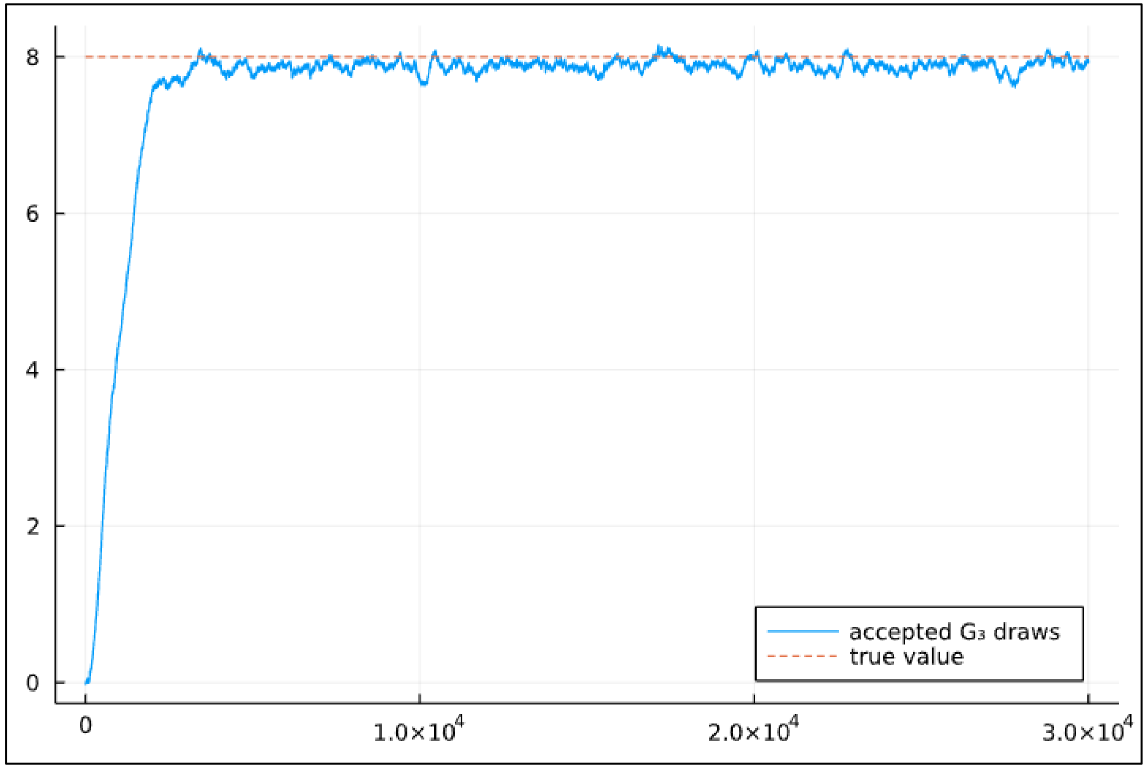
Accepted Parameter Draws from Bayesian Posterior



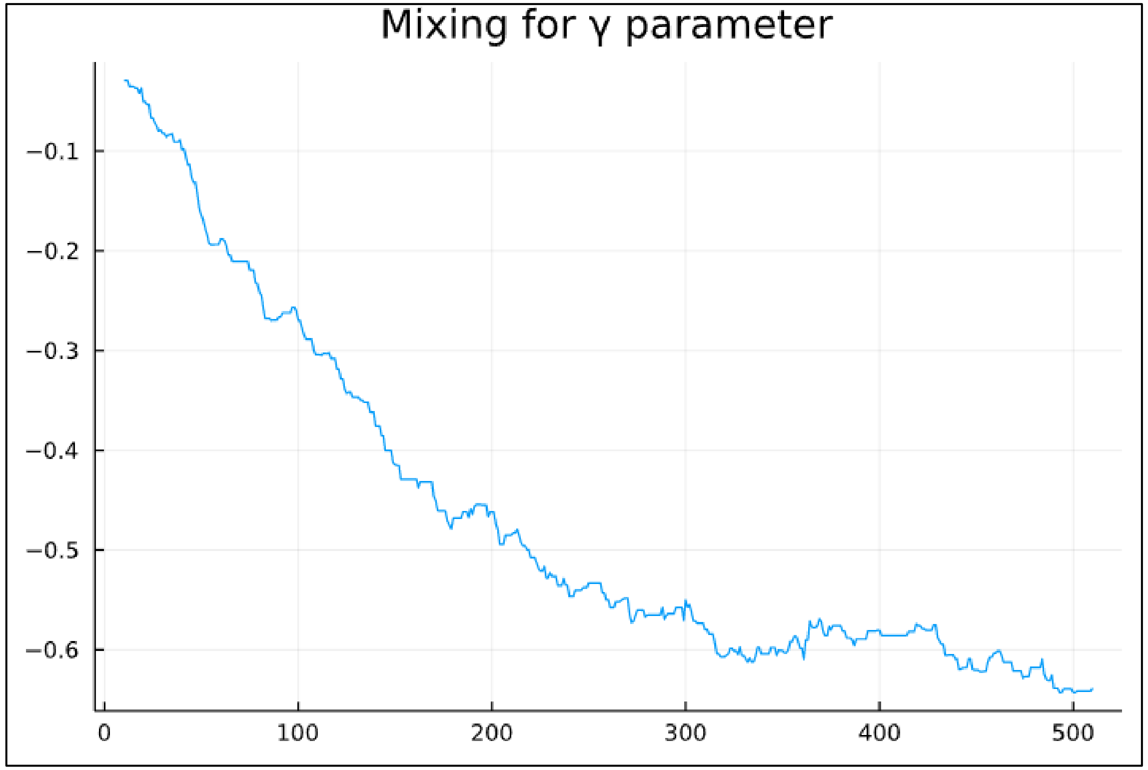


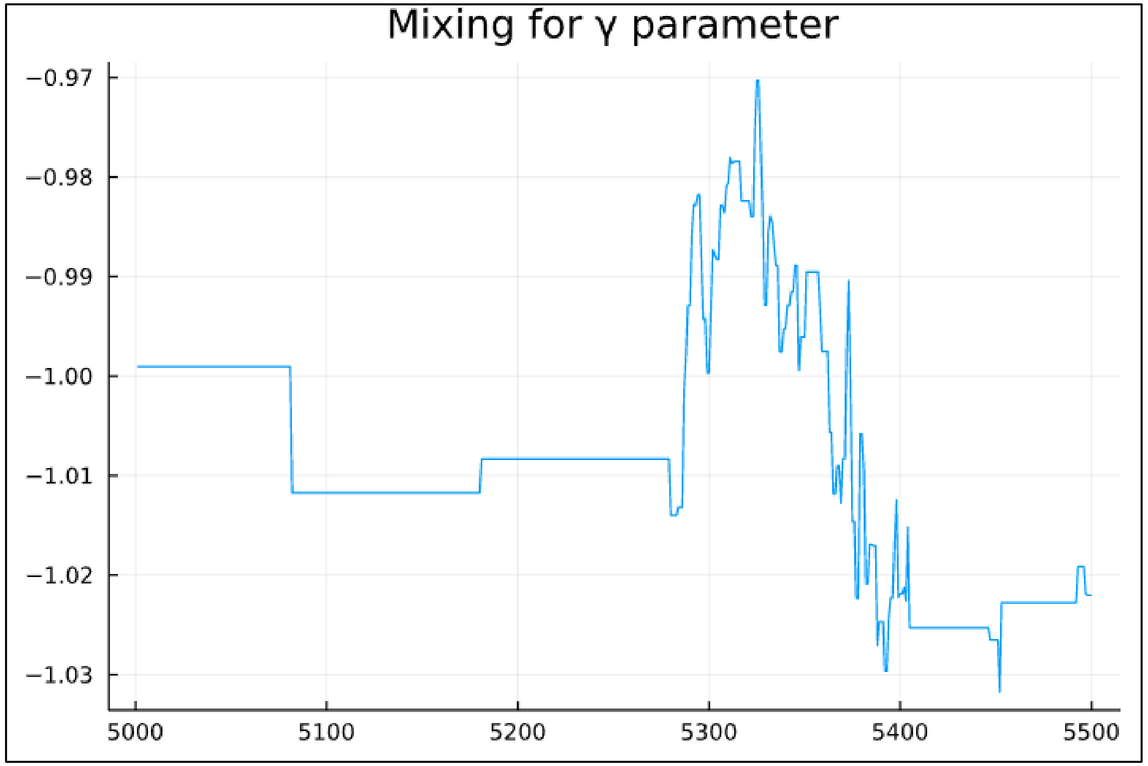
Accepted Parameter Draws from Bayesian Posterior



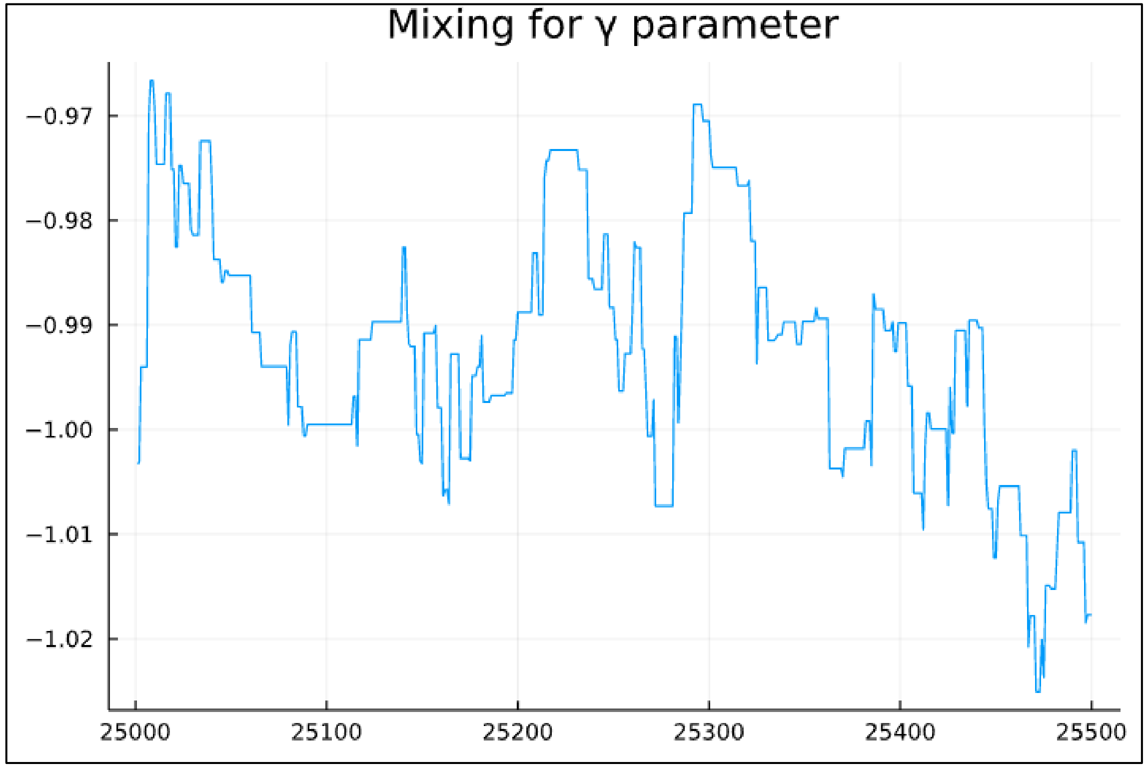


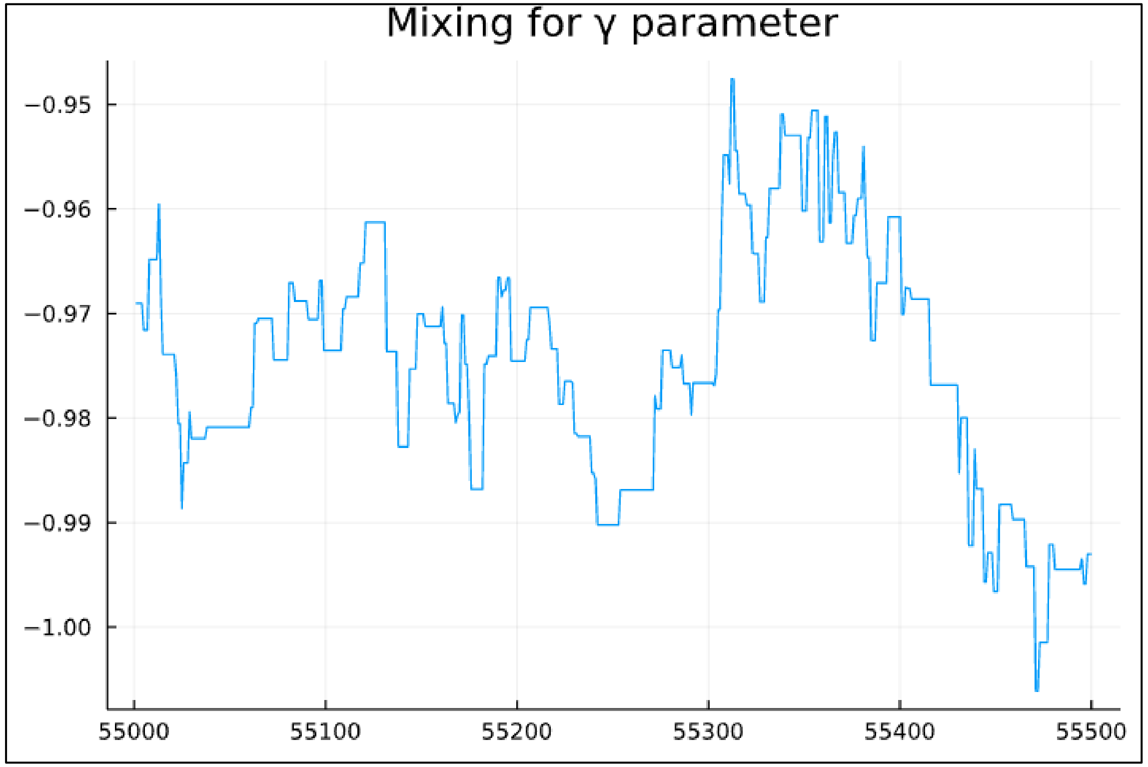
Parameter Draw Mixing During Burn-In



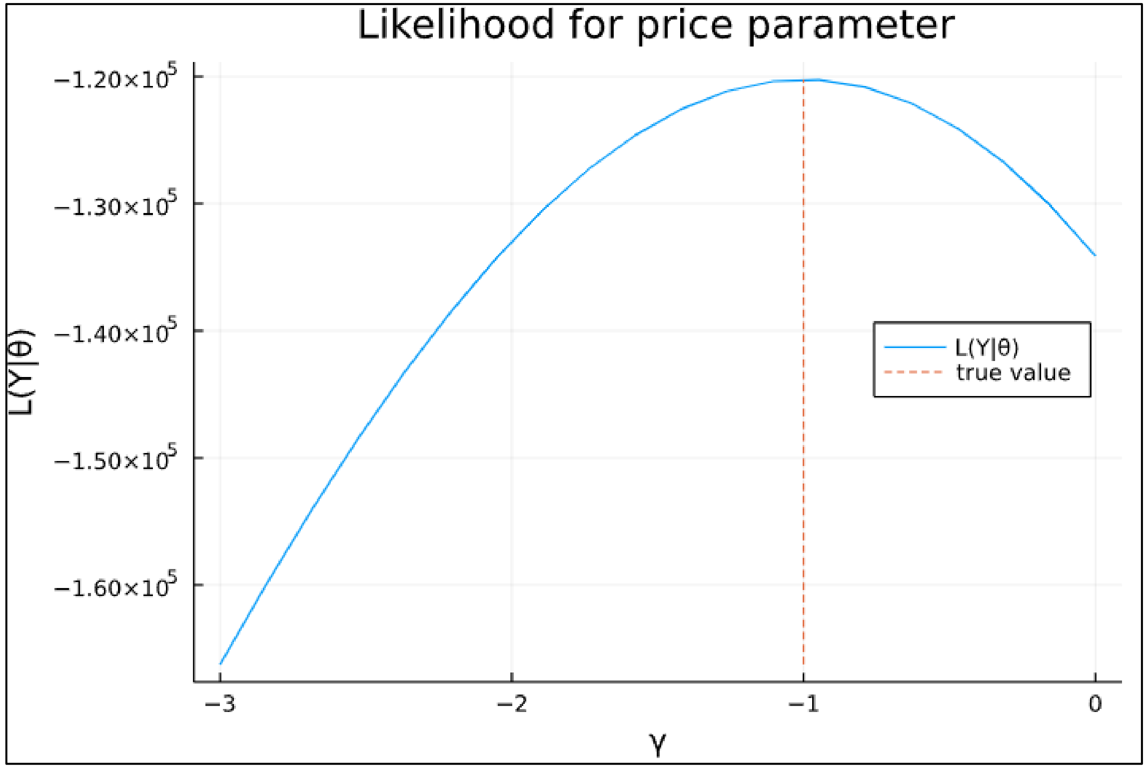


Parameter Draw Mixing After Burn-In

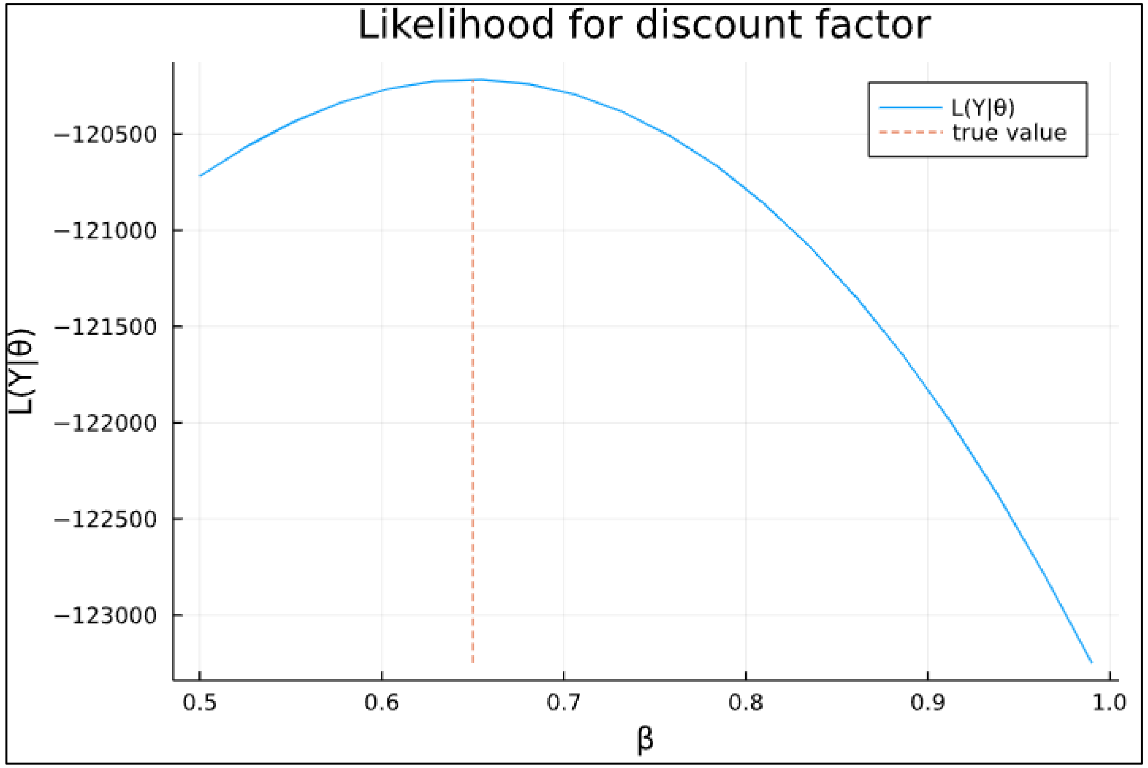




Sample Log Likelihood Function



Sample Log Likelihood Function



Heterogenous Case

Model Parameters and Environmental Variables

```
# ------------------------------------------------------------------------------
# 0. Parameters and setup.
# ------------------------------------------------------------------------------
# Model parameters (to be esimated).
α   = [-0.0, -0.0, -0.0]              # store brand intercepts
γ   = -1.0                            # price coefficient
G   = [1.0, 3.0, 8.0]                 # value of gifts
σg  = diagm([0.5, 1.0, 2.0])          # covariance of gift values (heterogeneous consumers)
β   = 0.650                           # discount rate

# Environment variables.
n_stores = 3                          # number of stores
n_choices = n_stores + 1              # number of choices for consumer (stores + outside option)
s̄ = [4, 4, 6]                         # gift threshold
price_mean = [1.0, 0.75, 1.5]         # mean of observed prices
price_stdev = diagm([0.25, 0.25, 0.25]) # standard deviation (covariance matrix) of observed prices
n_price_draws = 100                   # number of draws for price integration

# Simulated data parameters.
n_individuals = 100
n_periods = 1000

# Take draws for individual-level parameters
Gᵢ = rand(MvNormal(G, σg), n_individuals)
```

Parameter Estimates

Row	variable Any	true value Any	estimated mean Any	95% Credible Interval Low Any	95% Credible Interval High Any
1	α_1	-0.0	-0.036922	-0.0876735	0.0119379
2	α_2	-0.0	-0.0379751	-0.0778946	0.00114623
3	α_3	-0.0	-0.0632642	-0.13365	0.00545934
4	γ	-1.0	-0.970838	-1.01471	-0.925212
5	β	0.65	0.658644	0.648444	0.668853
6	$\bar{G}_1$	1.0	1.12314	0.967387	1.28021
7	$\bar{G}_2$	3.0	3.04384	2.81318	3.27425
8	$\bar{G}_3$	8.0	7.93703	7.62853	8.25045
9	σg_1	0.5	0.537511	0.394034	0.7284
10	σg_2	1.0	1.22921	0.908963	1.64703
11	σg_3	2.0	1.83767	1.29399	2.55138
12	G_2-10	4.07	3.52406	2.97044	4.10188
13	G_2-50	3.91	3.99739	3.40701	4.60856
14	G_2-100	3.62	3.67842	3.11153	4.24063

```
completed 30000 draws, homogeneous parameter acceptance rate 0.27
heterogeneous: [0.32 0.31 0.31 0.33 0.34 0.34 0.31 0.35 0.33 0.33]
```

Initial Parameter Guesses

Variable	Parameter Guess			
	True Value	Informed	Directional	Zeros
α_1	0	0	0	0
α_2	0	0	0	0
α_3	0	0	0	0
γ	-1	-1	-1	0
β	0.65	0	0	0
G_1	1	4	1	0
G_2	3	4	1	0
G_3	8	4	1	0
α_1	0.5	1	1	1
α_2	1	1	1	1
α_3	2	1	1	1

I experimented with different starting parameter values.

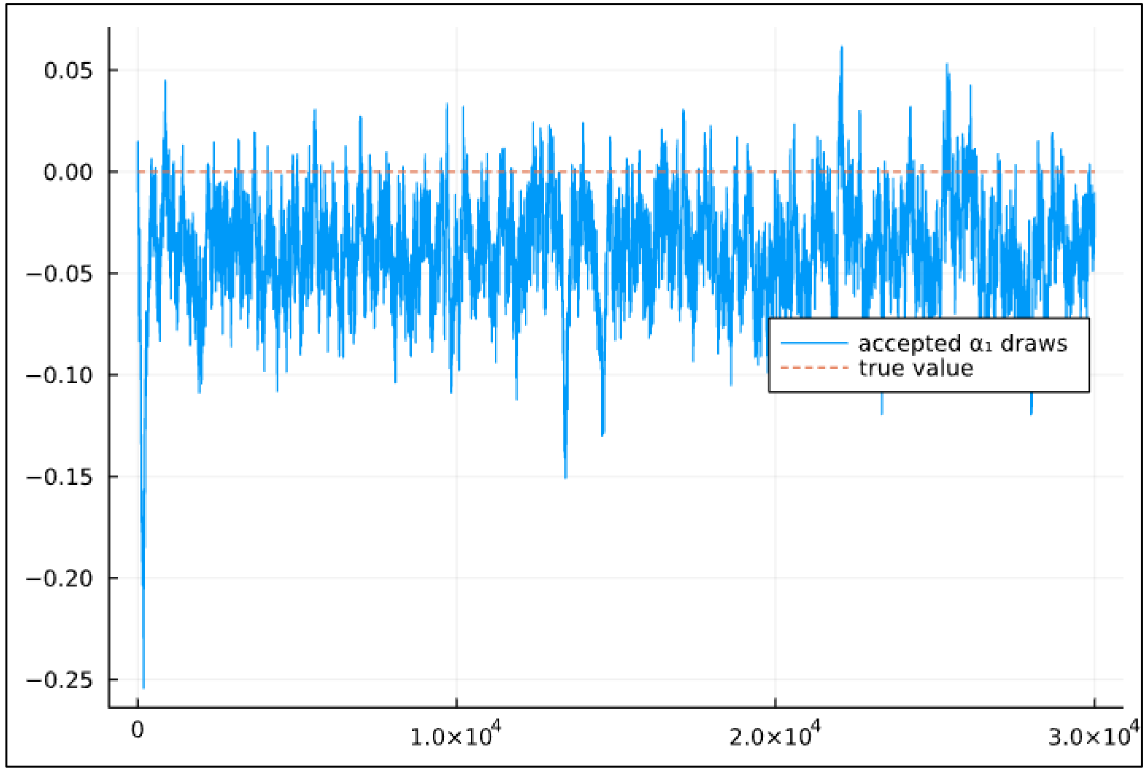
- Starting from the true value, to confirm that the distribution converged around these values.
- Informed and directional guesses which were logical choices based on economic theory and preliminary runs.
- A naïve guess of 0 for each parameter.

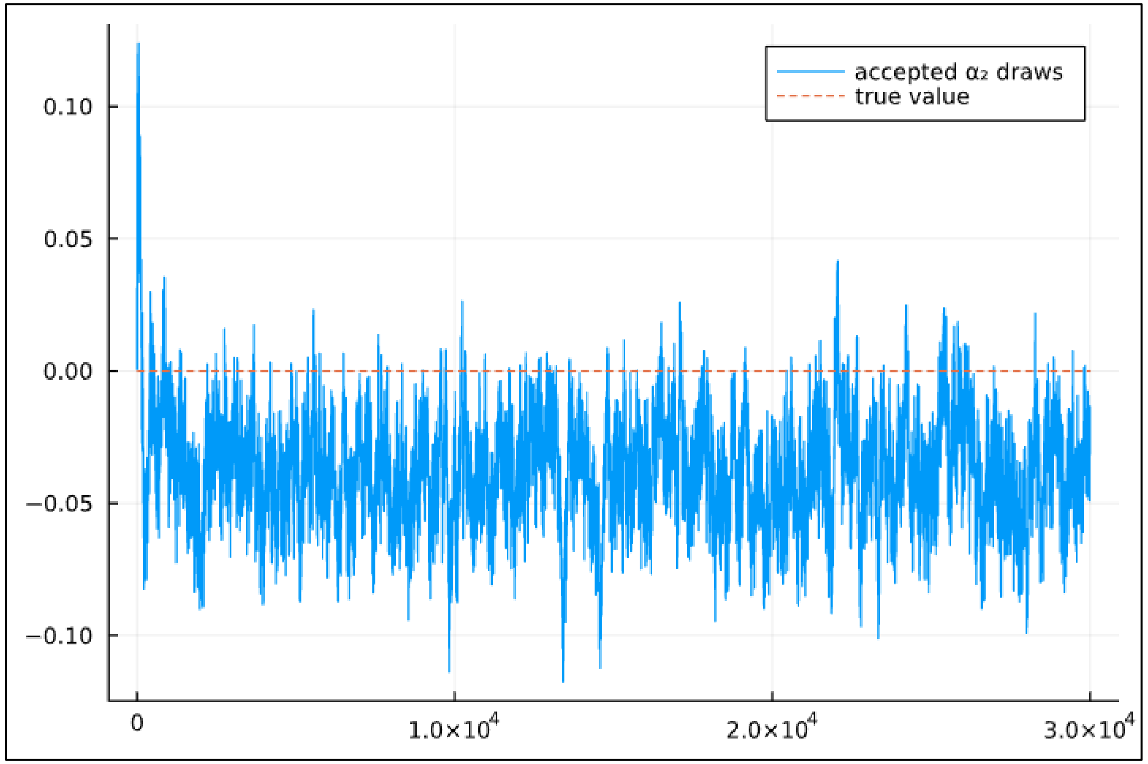
The distribution converged well before the burn-in period for all parameter guesses.

The following graphs and results all use the Zeros parameter guess.

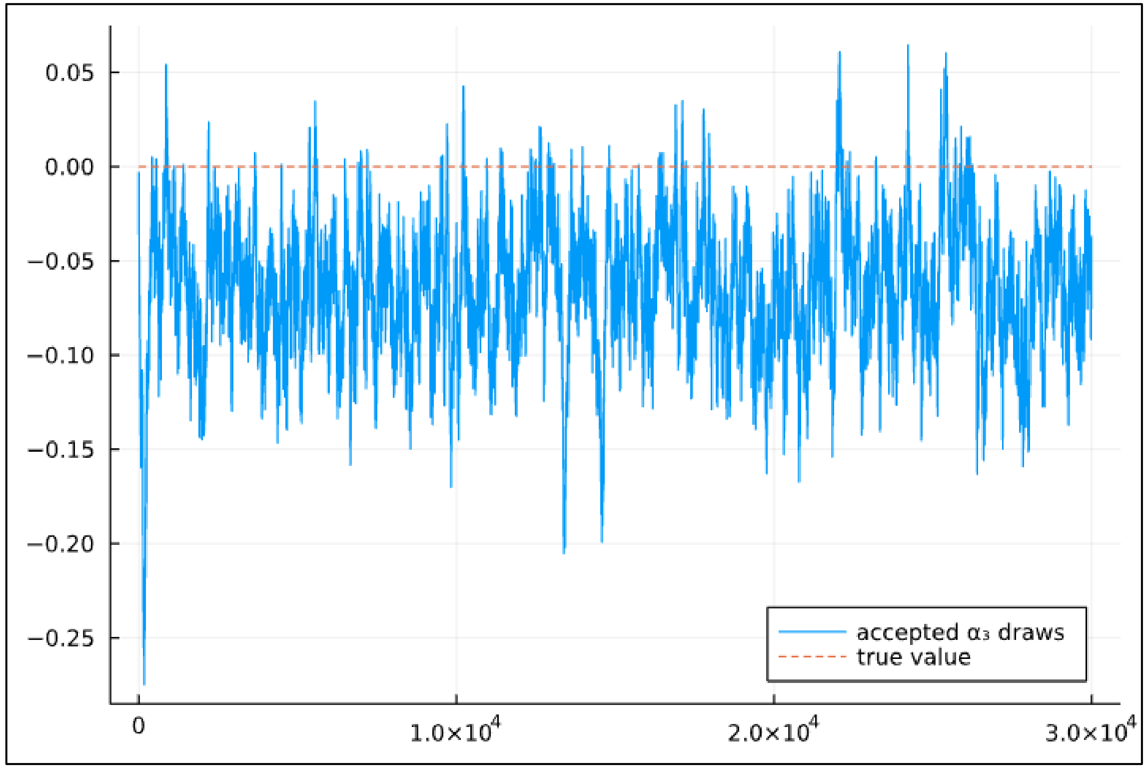
Heterogenous Case

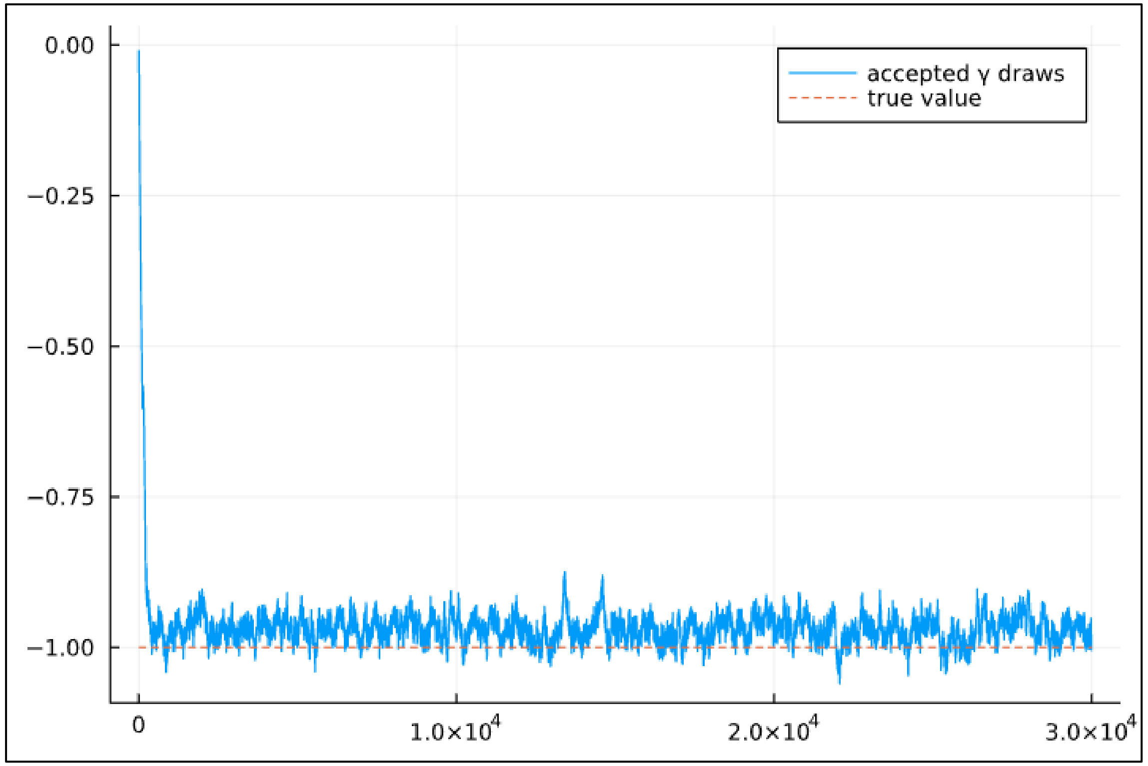
Accepted Parameter Draws from Bayesian Posterior



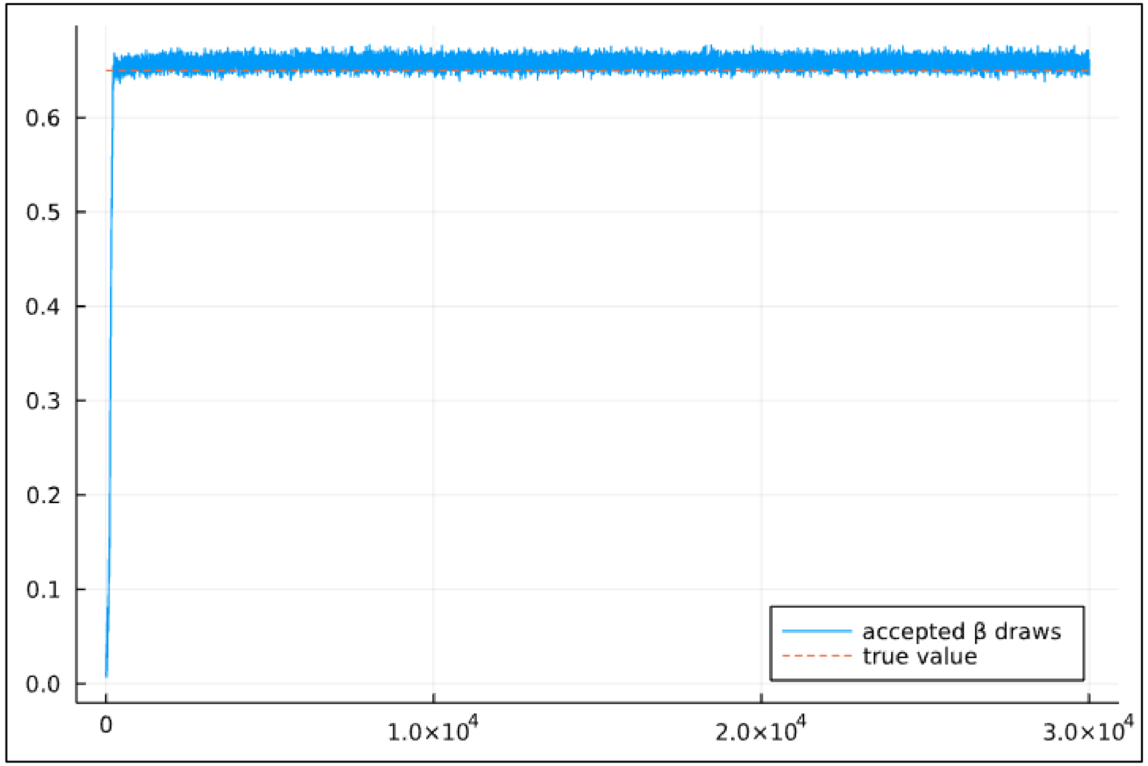


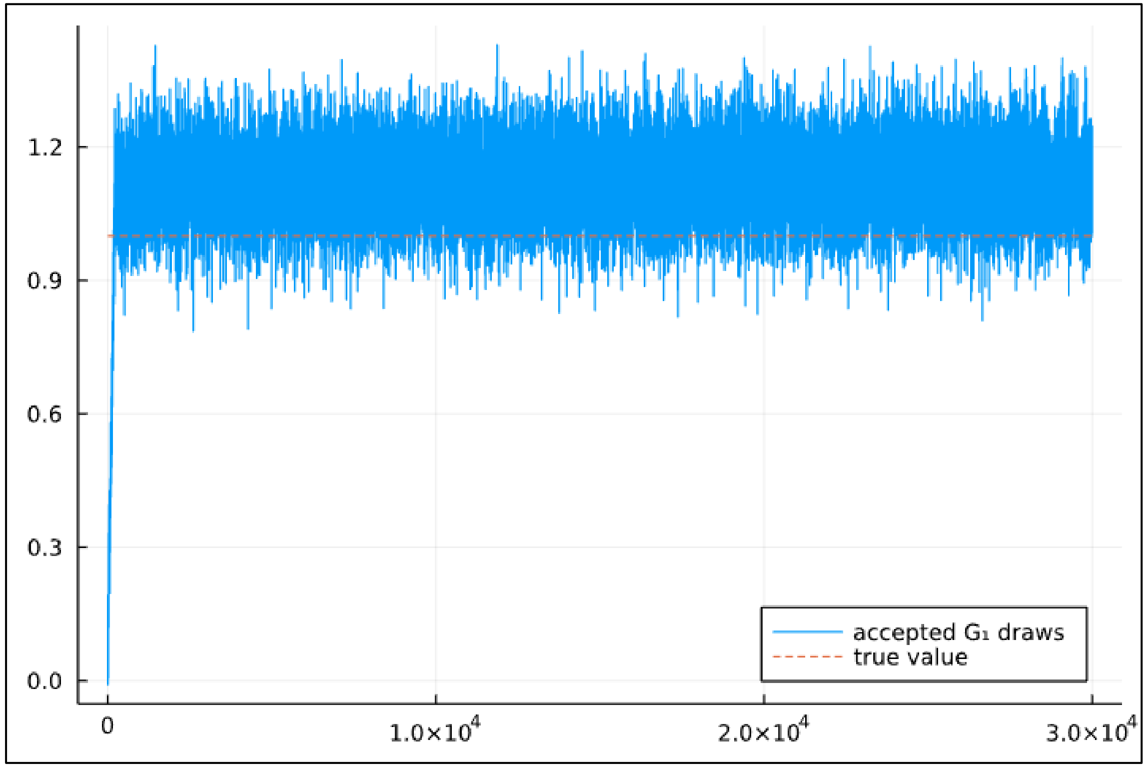
Accepted Parameter Draws from Bayesian Posterior



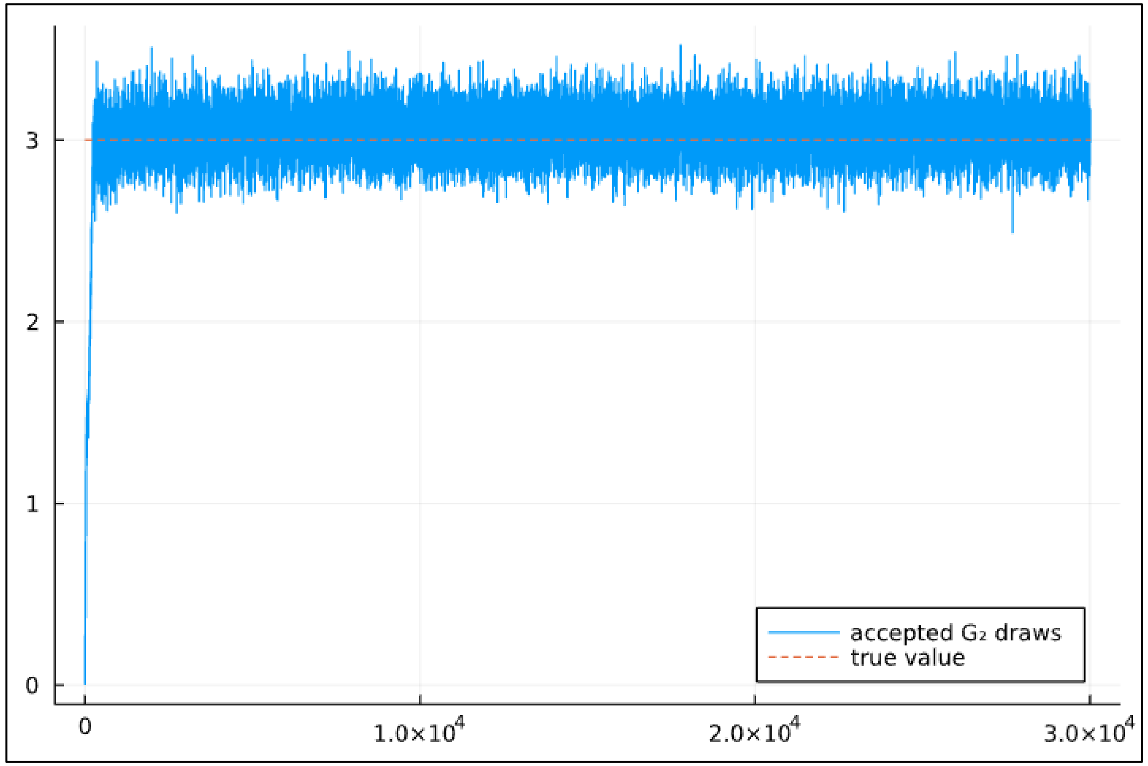


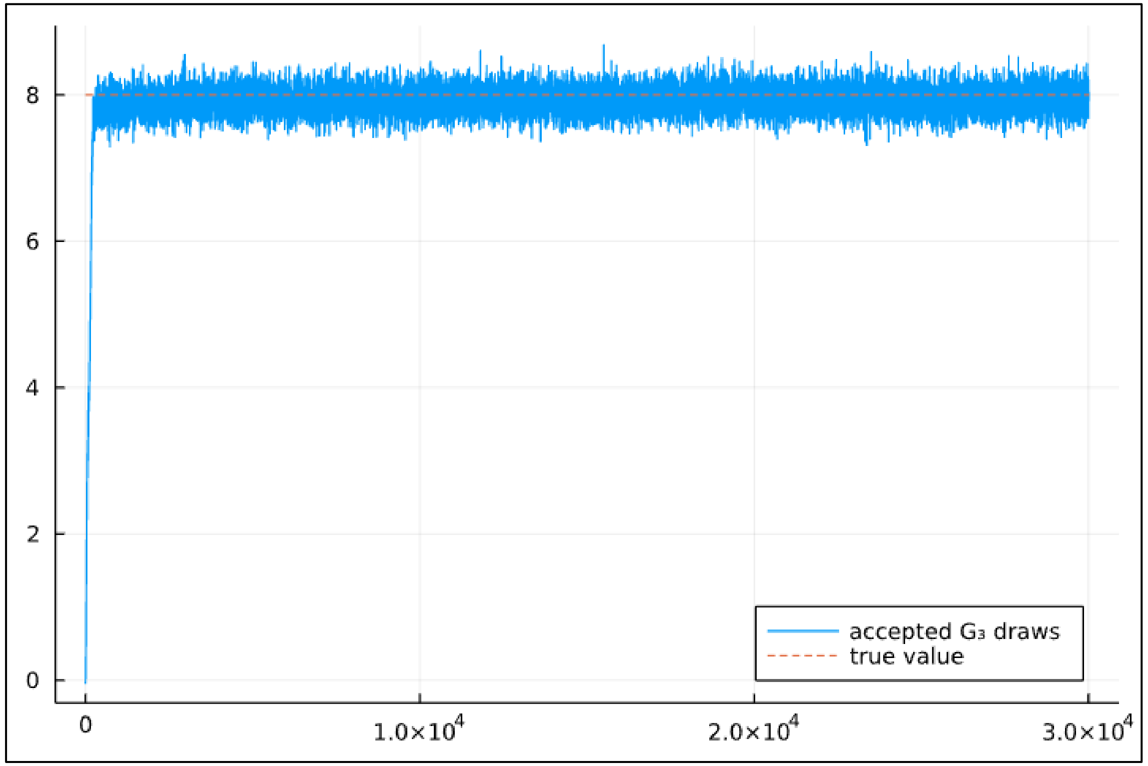
Accepted Parameter Draws from Bayesian Posterior



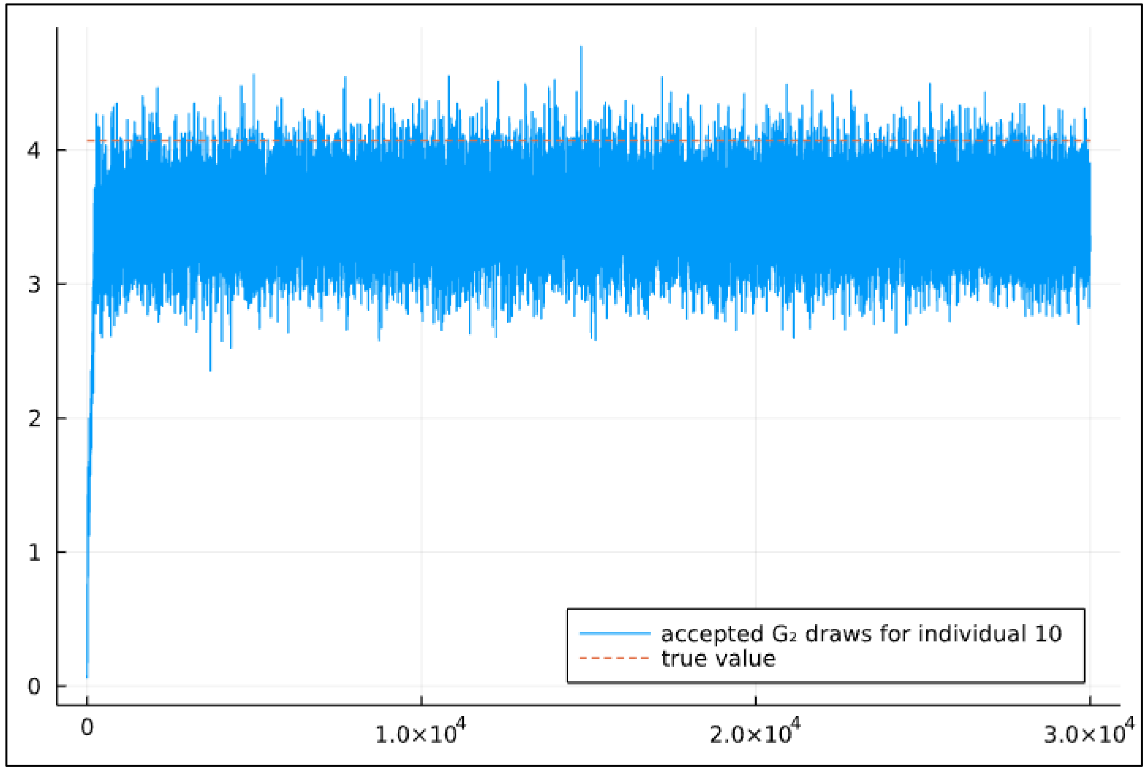


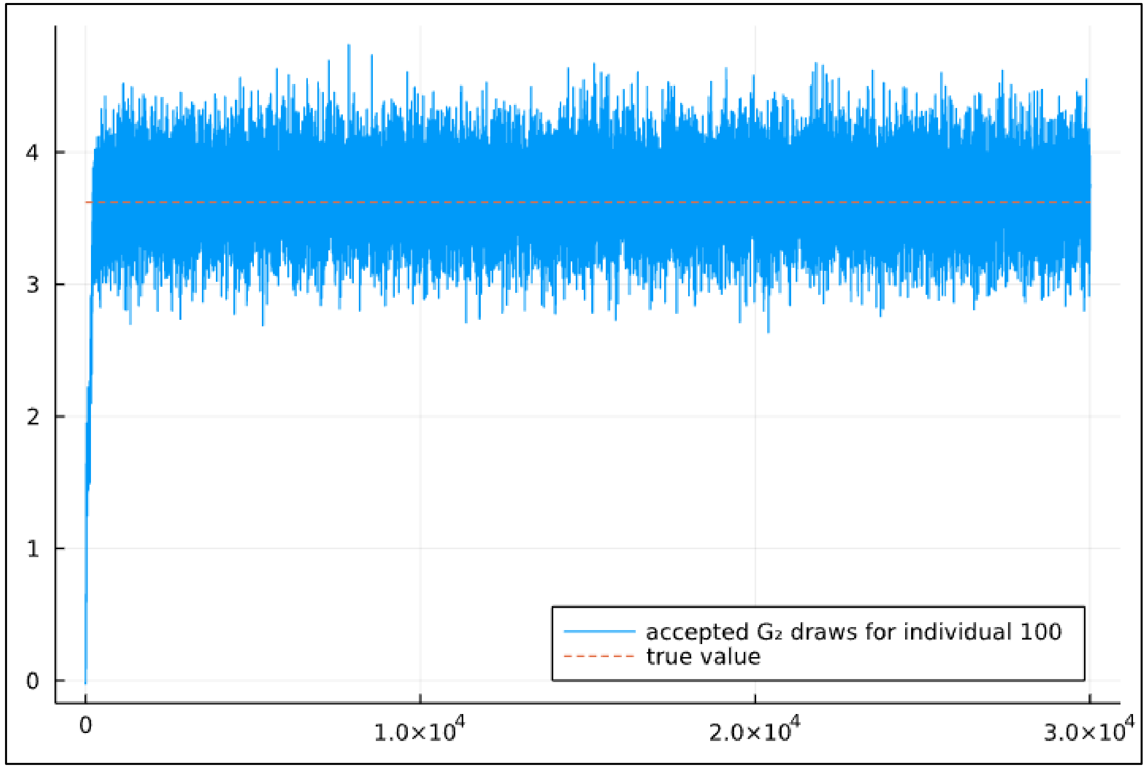
Accepted Parameter Draws from Bayesian Posterior





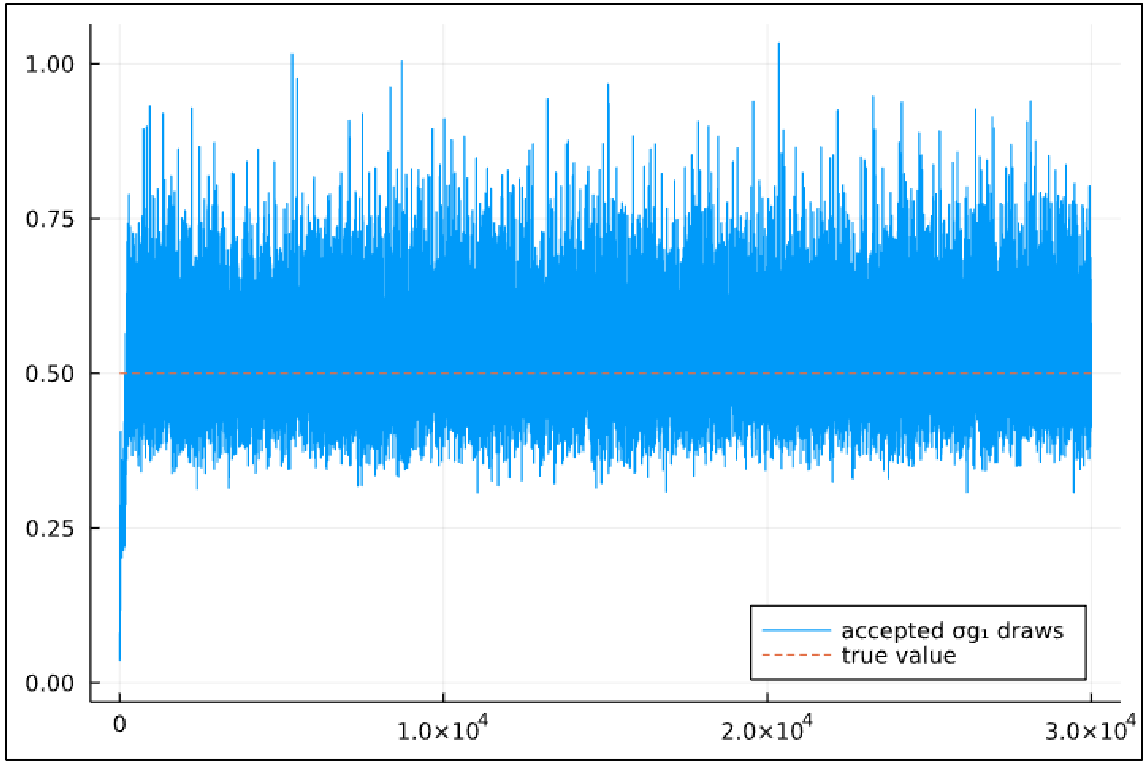
Accepted Parameter Draws from Bayesian Posterior

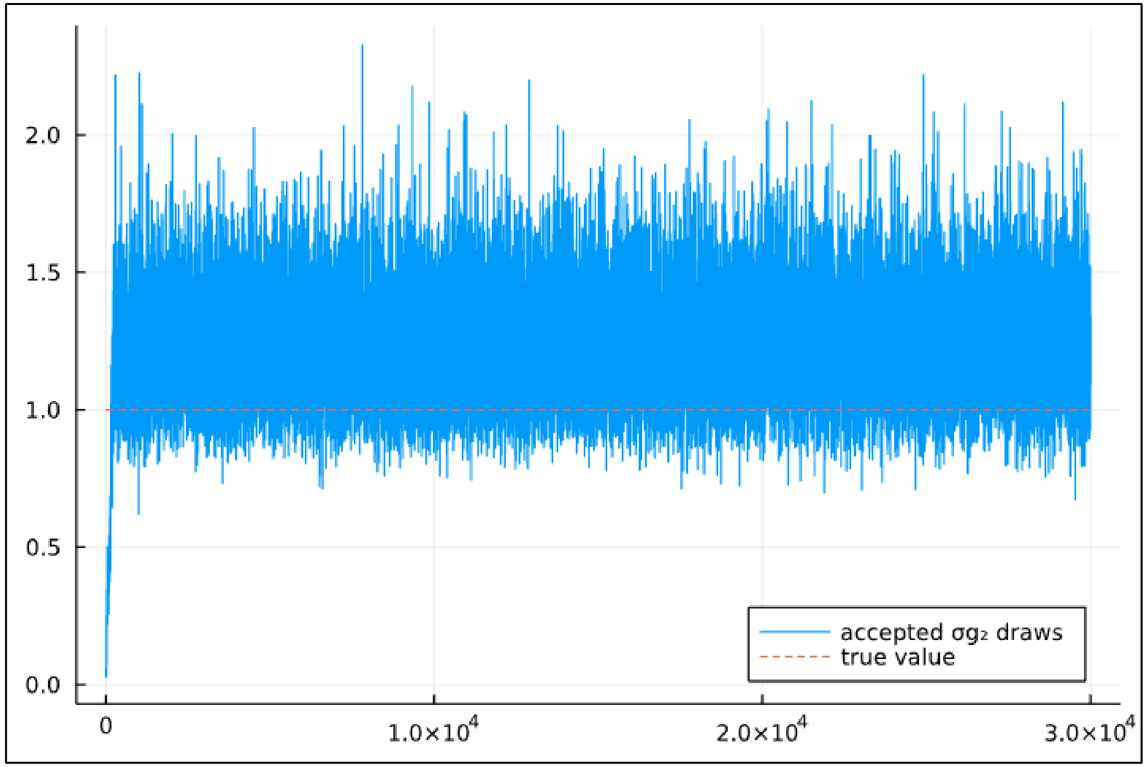




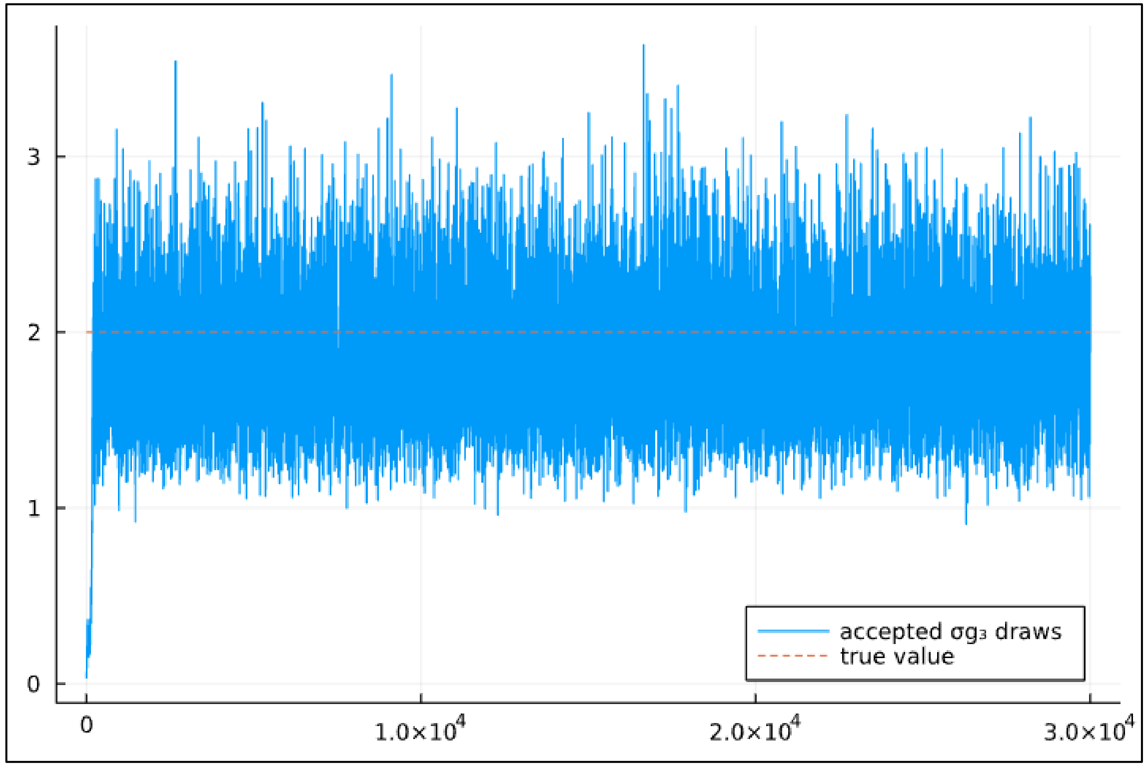
Heterogenous Case

Sample Log Likelihood Function

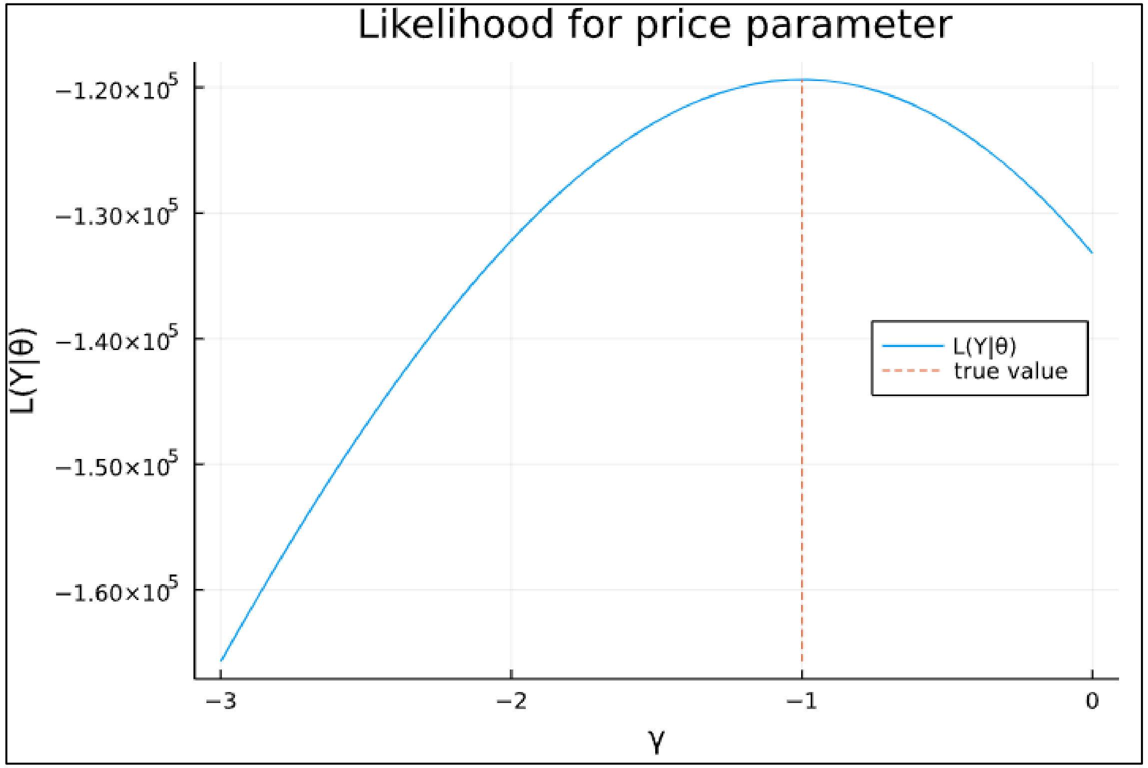


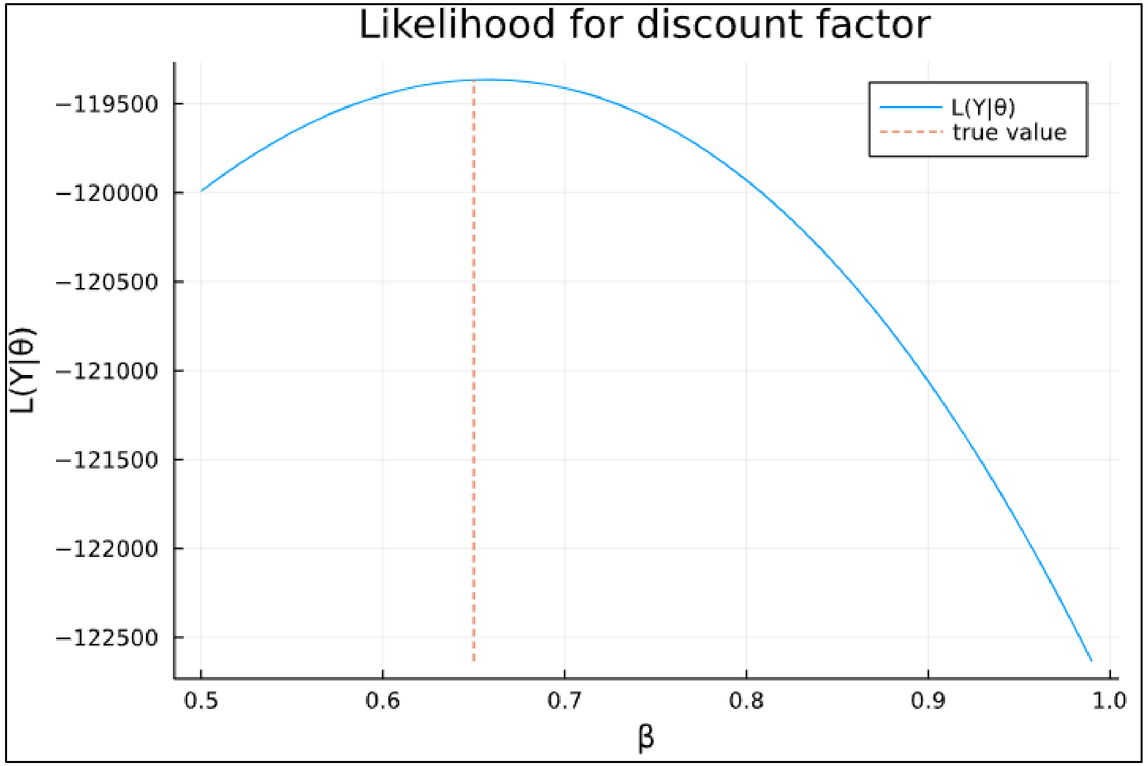


Sample Log Likelihood Function

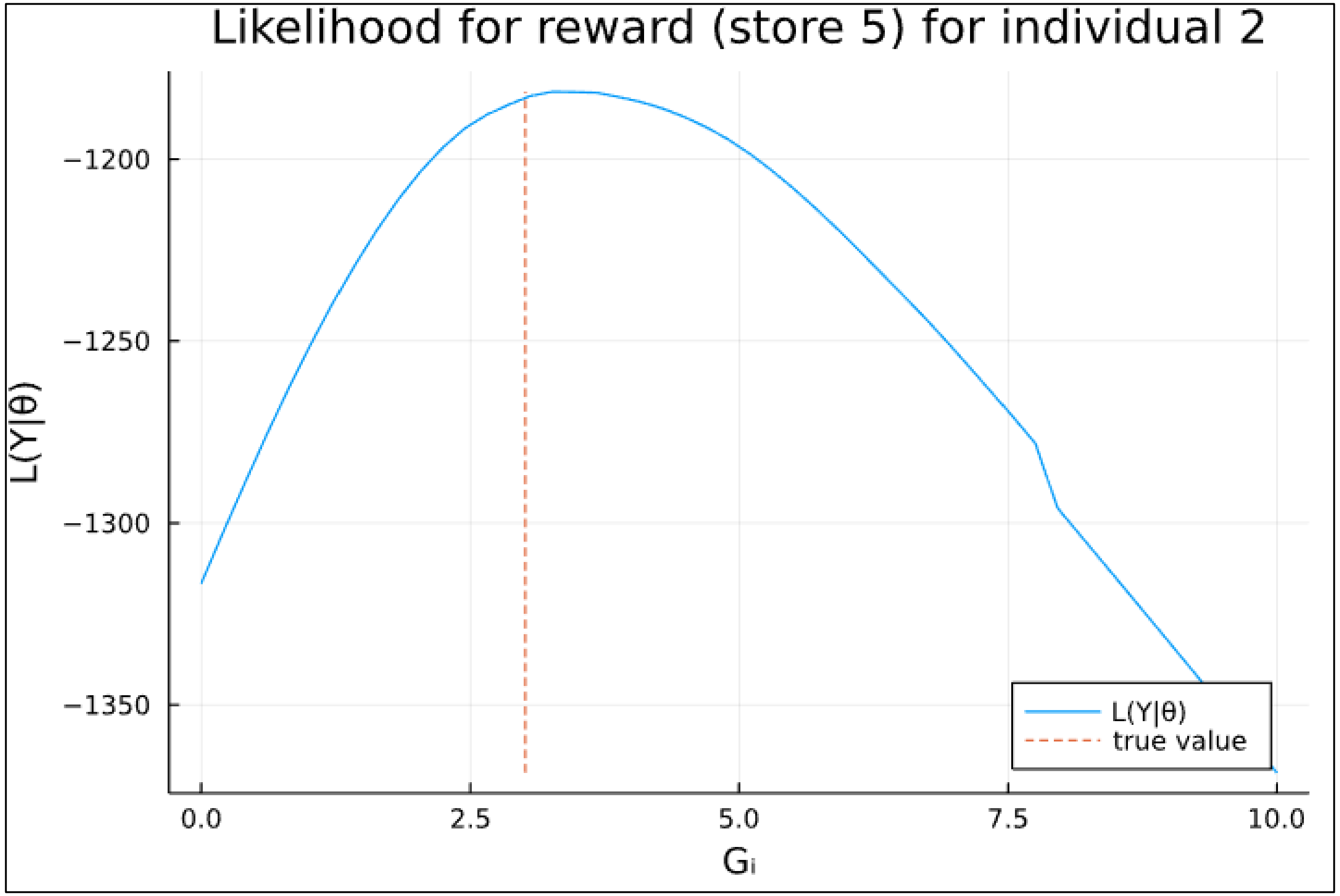


Sample Log Likelihood Function





Sample Log Likelihood Function



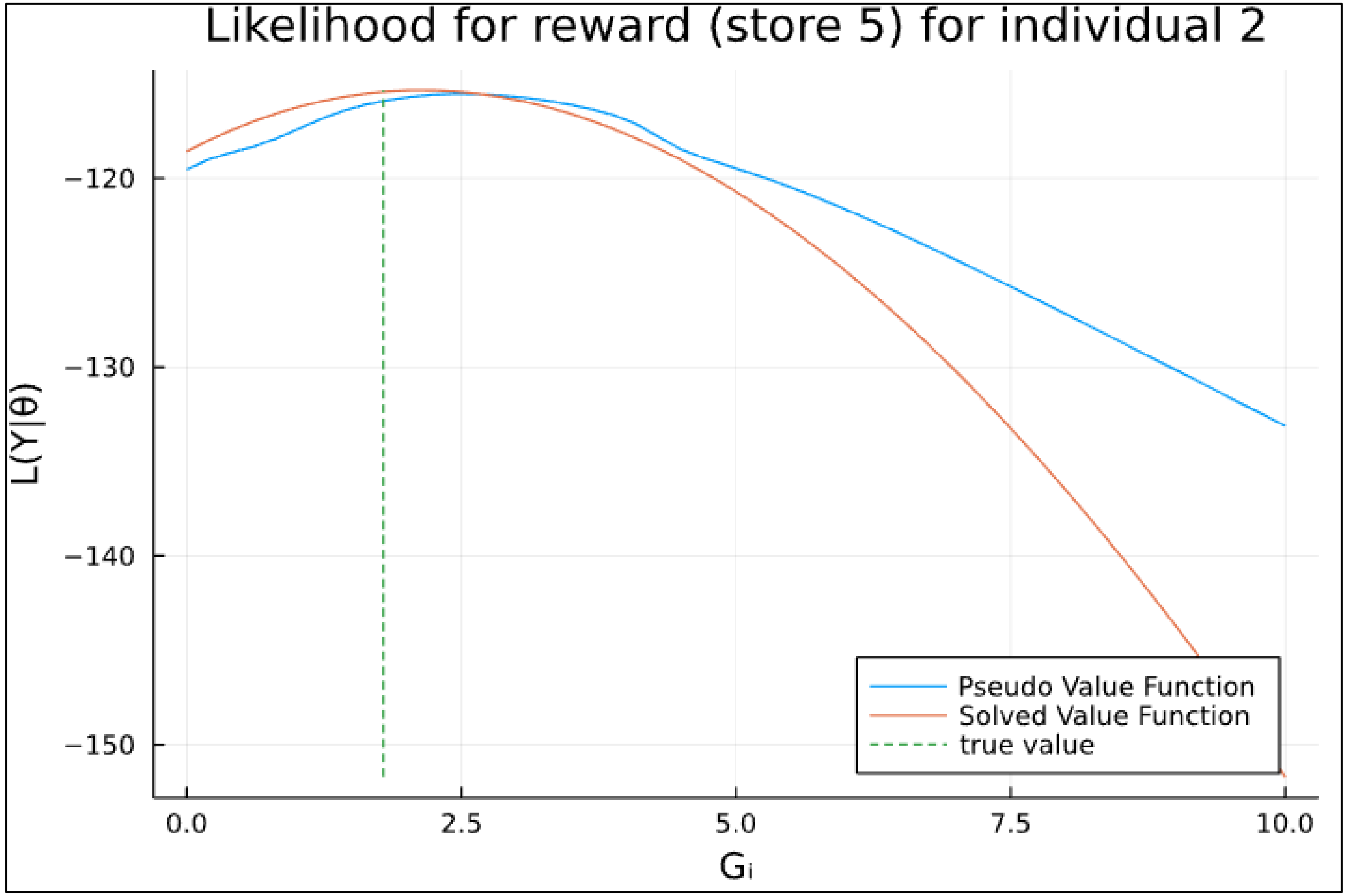
Issue with the Pseudo Value Function Approximation

The pseudo value function approximation to the true value function leads the likelihood function to be non-smooth and biased towards the population mean.

The bias towards the population mean causes incorrect estimates for the variance parameters.

The smoothness can be addressed by adjusting the kernel bandwidth parameter.

The bias can be addressed by using more periods (such as 1000 instead of 100) and by storing individual pseudo value function approximations for each individual (instead of the combined value function approach that stores a random individual in each period).



Outline of Code Files

File	Key sections	Notes
main.jl	0. Parameters and setup 1. Simulate data 2. Estimate model using IJC algorithm 3. Display results 4. Estimate model using Maximum Likelihood 5. Benchmarking, Profiling, Visualizing	Sections 0-3 perform the key steps of creating the dataset, drawing from the posterior using the IJC algorithm, and displaying the results. Section 4 is only present in the homogeneous case and is shown only for comparison. Section 5 contains additional information used for debugging and optimization.
custom_functions.jl	1. Precalculate_State_ Correspondence 2. Solve_Individual_Problem 3. Solve_Value_Function 4. Get_Proposal 5. Parameter_Distance_Kernel 6. Log_Prior_Individual_Params 7. Log_Sample_Likelihood 8. Get_Pseudo_Value_Function	Function 1 precalculates which states are accessible from each starting state and the corresponding choices and rewards. Function 2 is used to perform one iteration of value function iteration. Functions 4-8 are used when taking parameter draws.

File	Key sections	Notes
data_structs.jl	Model_Parameters_Struct Environment_Struct Dataset_Struct Value_Function_Parameters_Struct	Immutable data structs to store scalars and arrays used by various functions and steps in the main program. Model parameters refers to the key unobservables: α, γ, β etc. Environment struct contains observable model parameters such as the number of stores and the number of visits to earn a reward. Dataset struct contains the simulated dataset; store choices, starting states, and prices. The value function struct contains information required to calculate the pseudo value function; past value functions and corresponding parameter draws.

Optimization

The majority of compute time is spent on unavoidable log and exponential calculations.

- The functions used to update the value function (Solve_Individual_Problem) and calculate the sample log likelihood (Log_Sample_Likelihood) were the computational bottlenecks in the Homogeneous Case.
- Both functions were optimized such that they were bottlenecked by calls to the log and exponential functions.
- The Heterogeneous Case was bottlenecked by estimating the value function (Get_Pseudo_Value_Function). This function was further optimized in this set of code to also be limited by basic math operations.
- The fact that estimating the value function becomes the bottleneck illustrates the utility of the IJC Method, especially in the case of heterogeneous individuals.

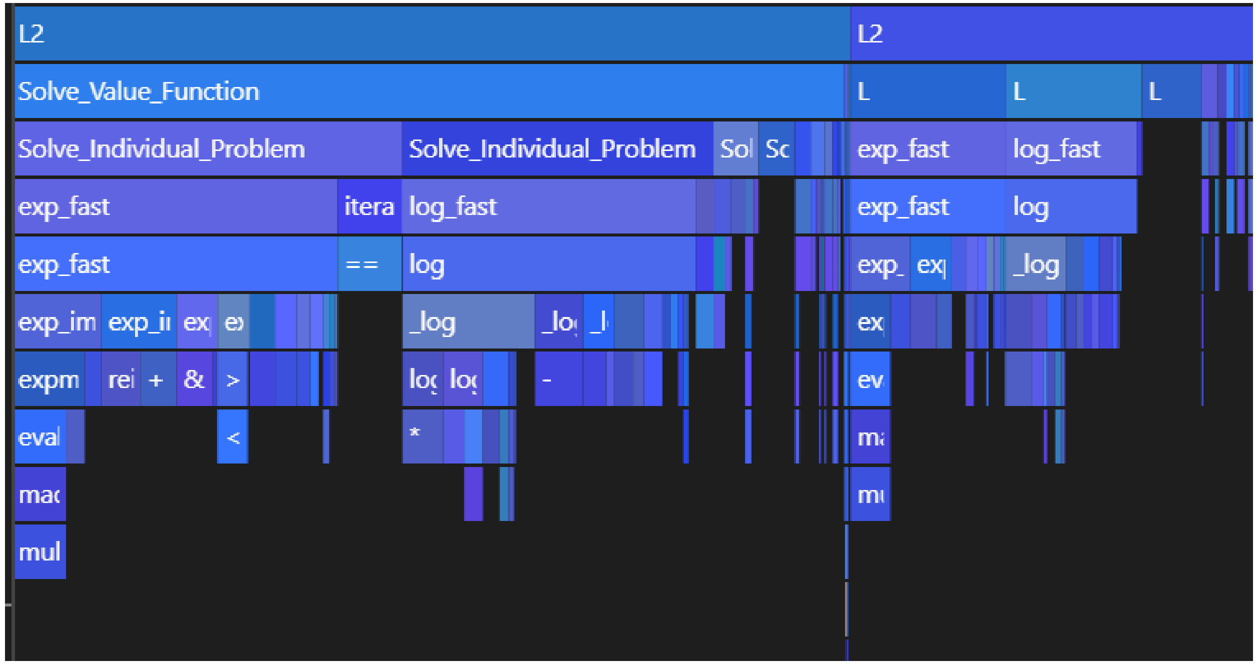
Parallelization is employed in the Heterogenous Case to greatly improve the speed of the computation bottleneck (calculating the pseudo value function).

- Calculating the log likelihood can be parallelized over individuals for a given parameter draw.
- Parallelization did not increase the speed of the Homogeneous Case. Calculating the likelihood for each individual did not require estimating the pseudo value function which resulted in the overhead of multithreading causing a net loss in performance.

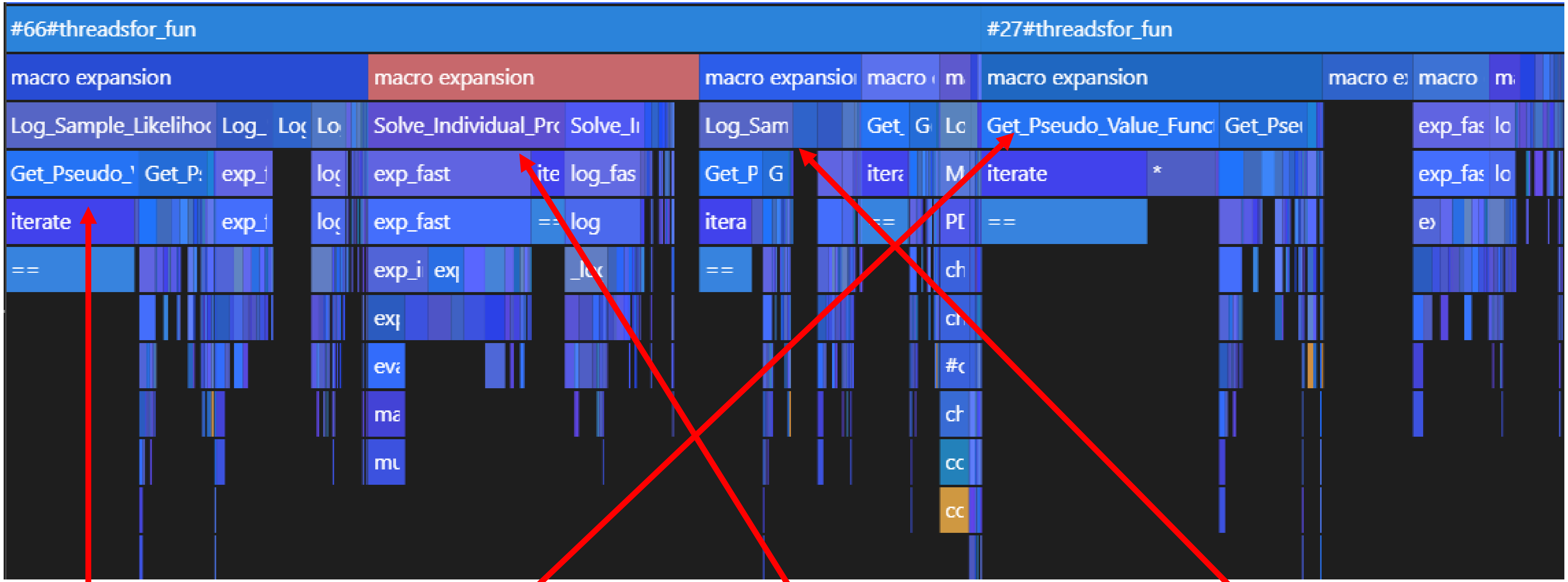
Optimization

Model	Dataset	Method	Compute Time
Homogeneous individuals	1000 individuals 100 periods	IJC method - Bayesian MCMC	15 minutes
		Bayesian MCMC without IJC (solving value function instead)	54 minutes
		Maximum Likelihood with numerical gradients	2.5 minutes
Heterogeneous individuals	1000 individuals 100 periods	IJC method - Bayesian MCMC	290 minutes
		Bayesian MCMC without IJC (solving value function instead)	792 minutes
	100 individuals 1000 periods	IJC method - Bayesian MCMC	23 minutes
		Bayesian MCMC without IJC (solving value function instead)	786 minutes

Optimization



Profiling (IJC method, individual value functions)



Get_Pseudo_Value_Function: 36%

Solve_Individual_Problem: 15%

Log_Likelihood: 10%